1 Appendix

Note: The LSGM parameter set, \( \Phi \), which appears in the Main Document is omitted here for conciseness.

1.1 Joint distribution

The joint distribution of an observation sequence \( X_{1:T} \) and a state sequence \( \Pi_{1:T} \) is given by the following expression. Equation 2 uses the LSGM independence assumptions.

\[
\log P[X_{1:T}, \Pi_{1:T}] = \sum_{t=1}^{T} \left( \log P[\pi_{t}|\Pi_{1:t-1}, X_{1:t-1}] + \log P[x_{t}|\pi_{t}, \Pi_{1:t-1}, X_{1:t-1}] \right) = \\
\sum_{t=1}^{T} \left( \log P[\pi_{t}|\pi_{t-1}, X_{1:t-1}] + \log P[x_{t}|\pi_{t}, X_{1:t-1}] \right) = \\
\sum_{t=1}^{T} \left( \log \gamma_{t-1}(\pi_{t-1}, \pi_{t}) + \log e_{t}(\pi_{t}) \right)
\]

1.2 Forward, backward, Viterbi recursions

The forward recursion is derived as follows

\[
e_{t+1}(\pi_{t+1}) \sum_{\pi_{t}} \alpha_{t}(\pi_{t}) \gamma_{t}(\pi_{t}, \pi_{t+1}) = \quad (4)
\]

\[
P[x_{t+1}|\pi_{t+1}, X_{1:t}] \sum_{\pi_{t}} P[X_{1:t}, \pi_{t}] P[\pi_{t+1}|X_{1:t}, \pi_{t}] = \\
\sum_{\pi_{t}} P[\pi_{t+1}|\pi_{t}, X_{1:t}] \sum_{\pi_{t}} P[X_{1:t}, \pi_{t}, \pi_{t+1}] = \\
P[X_{1:t+1}, \pi_{t+1}] = \alpha_{t+1}(\pi_{t+1})
\]

The backward recursion is derived as follows (Equation 11 uses the LSGM independence assumptions)

\[
\sum_{\pi_{t+1}} \beta_{t+1}(\pi_{t+1}) \gamma_{t}(\pi_{t}, \pi_{t+1}) e_{t+1}(\pi_{t+1}) = \quad (8)
\]

\[
\sum_{\pi_{t+1}} P[X_{t+2:T}|\pi_{t+1}, X_{1:t+1}] P[\pi_{t+1}|\pi_{t}, X_{1:t}] P[x_{t+1}|\pi_{t+1}, X_{1:t+1}] = \\
\sum_{\pi_{t+1}} P[X_{t+1:T}|\pi_{t+1}, X_{1:t+1}] P[\pi_{t+1}|\pi_{t}, X_{1:t+1}] = \\
\sum_{\pi_{t+1}} P[X_{t+1:T}|\pi_{t+1}, \pi_{t}, X_{1:t}] P[\pi_{t+1}|\pi_{t}, X_{1:t}] = \\
\sum_{\pi_{t+1}} P[X_{t+1:T}, \pi_{t+1}|\pi_{t}, X_{1:t}] = \\
P[X_{1:T}, \pi_{t}, X_{1:t}] = \beta_{t}(\pi_{t})
\]

The Viterbi recursion for the LSGM may be derived as follows. Equation 15 uses the LSGM independence assumptions.

\[
\max_{\pi_{t}} P_{\max}[X_{1:t+1}, \pi_{t+1}] = \\
\max_{\pi_{t}} P_{\max}[X_{1:t}, \pi_{t}] P[\pi_{t+1}|\pi_{t}, X_{1:t}] P[x_{t+1}|\pi_{t}, X_{1:t}, \pi_{t+1}] = \\
\max_{\pi_{t}} P_{\max}[X_{1:t}, \pi_{t}] P[\pi_{t+1}|\pi_{t}, X_{1:t}] P[x_{t+1}|\pi_{t+1}, X_{1:t}]
\]
Defining the Viterbi recursion variable as \( \tau_t(k) = P_{\text{max}}[X_{1:t}, \pi_t = k] \), and locating the transition and emission terms in Equation 15, we obtain

\[
\tau_{t+1}(k) = \max_j \tau_t(j) \gamma_t(j, k)e_{t+1}(k)
\]  

(16)

1.3 Modified forward recursion for length &gt; 1 emissions

For a state emitting \( l \)-symbols at once (such as the motif state), the following recursion replaces the standard forward recursion. Here we assume that the state \( \pi_i \) emits symbol \( x_i \), as before, however the state may emit more than one symbol as a group - in which case \( x_i \) is the last symbol emitted as a group.

\[
P[X_1:i, \pi_i = j] = \sum_{\pi_{i-l}} P[X_{1:i-l}, \pi_{i-l}] P[\pi_i = j | \pi_{i-l}, X_{1:i-l}]
\]

\[
P[X_{i-l+1:i} | \pi_i, X_{1:i-l}] = \sum_{\pi_{i-l}} P[X_{1:i-l}, \pi_{i-l}] P[\pi_i | \pi_{i-l}, X_{1:i-l}]
\]

(17)

\[
P[X_{i-l+1:i} | \pi_i] = \sum_{\pi_{i-l}} P[X_{1:i-l}, \pi_{i-l}] P[\pi_i | \pi_{i-l}, X_{1:i-l}]
\]

(18)

Equation 17 gives the modified forward recursion for the first-order full LSGM. Equation 18 provides the recursion for a half LSGM such as used in our TF-binding experiments.

1.4 Additional details

1.4.1 Multidimensional synthetic data experiments

The HMM has 3 states each emitting 4-dimensional full-covariance Gaussians. This requires \( 3 \times (4^2 + 4) = 60 \) parameters. In addition, it uses 6 transition probabilities and 2 start probabilities bringing the total parameter count to 68.

The LSGM uses 65 parameters which are provisioned as follows.

- 1 GRU layer with 1 neuron without bias requiring \( 3 \times (4 \times 1 + 1^2) = 15 \) parameters. A GRU layer without bias needs \( 3(mn + n^2) \) parameters for \( m \) inputs \( n \) outputs (as implemented in PyTorch version 0.4).

- 1 dense neuron taking the GRU output as input and providing a single output. Requires \( 1 \times 1 + 1 = 2 \) parameters.

- Emission layer accepting the dense neuron’s output and providing means and diagonal covariances for 3 states. The covariances use a single bias value per state. This requires \( 1 \times 12 + 12 \) parameters for mean and \( 1 \times 12 + 3 \) parameters for variance for a total of 39 parameters

- Transition layer accepting the dense neuron’s output and providing 9 outputs representing the transition matrix. The transition network doesn’t use biases. This needs \( 1 \times 9 = 9 \) parameters

We performed the experiments over five times for each case, and took the average performance on the test set.

1.4.2 Multidimensional synthetic data generator parameters

- \( \bar{\mu}_1 = (0, 0, 0, 0)^T, \bar{\mu}_2 = (1, 1, 1, 1)^T, \bar{\mu}_3 = (-1, -1, -1, -1)^T \)

- \( \bar{\sigma}_1 = \bar{\sigma}_2 = \bar{\sigma}_3 = (0.01, 0.01, 0.01, 0.01) \)
• The diagonal covariance from $f$ is limited to between 0.01, 0.04 using sigmoid activation and scaling. $f$ uses 33 GRU units

• $a_{pq} = \{0.7, \text{ if } p = q, 0.15 \text{ otherwise}\}$

1.4.3 Synthetic scalar sequence generation

The algorithm for generating scalar synthetic sequences is given below. After this procedure is run, numpy (numpy.mean, numpy.var) is used to determine mean and standard deviation of all the datapoints in all the sequences, and then the sequences are normalized as $\frac{\text{synthetic} - \mu}{\sigma}$.

Algorithm 1 Synthetic scalar data generation

```python
1: procedure chooseTransition(runningSum)
2: \( \alpha = 10 \)
3: \( \beta = 30 \)
4: if runningSum < \( \alpha \) then
5: \( \vec{p} = [0.9, 0.05, 0.05]^T \)
6: end if
7: if \( \alpha \leq \text{runningSum} < \beta \) then
8: \( \vec{p} = [0.1, 0.8, 0.1]^T \)
9: end if
10: if \( \beta \leq \text{runningSum} \) then
11: \( \vec{p} = [0.05, 0.05, 0.9]^T \)
12: end if
13: return \( \vec{p} \)
14: end procedure

15: procedure Emit(seed, state)
16: \( K = 1.2 \)
17: if state = 0 then
18: return seed \times K
19: end if
20: if state = 1 then
21: return seed
22: end if
23: if state = 2 then
24: return seed/K
25: end if
26: end procedure

27: procedure genSyntheticData(Length)
28: synthetic = empty list
29: seed = random between 1e-2, 1
30: counter = 0
31: for 0 < counter < Length do
32: if counter > 0 then
33: \( r = \text{sum of last 10 items in synthetic} \)
34: \( \vec{p} = \text{chooseTransition}(r) \)
35: else
36: \( \vec{p} = [1/3, 1/3, 1/3]^T \)
37: end if
38: state = \text{Sample from multinomial distribution } \vec{p}
39: seed = \text{Emit(seed, state)}
40: Add seed to synthetic
41: end for
42: return synthetic
43: end procedure
```
1.4.4 Details of experiments with synthetic scalar sequences

We trained HMMs with 3 to 60 states on 1000 sequences of length 25 generated using the procedure above using Expectation Maximization, similar to the method for Multidimensional synthetic data. The model that achieved the best BIC had 52 states. This corresponds to $52 \times 51$ (transitions) + $52 \times 2$ (means, variances; emissions) + $51$ (start probability) = 2807 parameters. We selected this model and reran the experiment with the model five times that gave an average log-likelihood of 1.416. This value was relatively stable across the five experiments.

We trained a single LSGM model with 2 GRU units and 10 states following the same architecture as for the case of multidimensional synthetic data. This model has a total of only 284 parameters, almost a tenth the size of the HMM. The model gave an average log-likelihood per data point of 2.259 with a sample standard deviation of 0.662 over five runs. While the model seems to be sensitive to initialization, it also provided better performance on average with a significantly lower parameter count.

1.4.5 Details of architecture search for TF-binding experiment

The following architectures were searched for the TF-binding experiment.

Table 1: Model parameters explored

<table>
<thead>
<tr>
<th>Parameters</th>
<th>DeepBind</th>
<th>DeeperBind</th>
<th>LSGM</th>
<th>HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num motifs</td>
<td>5,16</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Motif length</td>
<td>11,24</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>LSTM layers</td>
<td>NA</td>
<td>1,2</td>
<td>2</td>
<td>NA</td>
</tr>
<tr>
<td>LSTM units</td>
<td>NA</td>
<td>20,32,64,128</td>
<td>64,128</td>
<td>NA</td>
</tr>
<tr>
<td>Dense layers</td>
<td>2</td>
<td>1,2</td>
<td>2</td>
<td>NA</td>
</tr>
<tr>
<td>Dense units</td>
<td>32</td>
<td>128,256</td>
<td>=LSTM units</td>
<td>NA</td>
</tr>
<tr>
<td>$F(x)$</td>
<td>NA</td>
<td>NA</td>
<td>$exp(x), (ReLU(x))^2, (ReLU(x))^4$</td>
<td>=LSGM</td>
</tr>
</tbody>
</table>

1.4.6 Music modeling LSGM architecture

![Figure: Music model architecture](image)

Table: Layer sizes in music model

**Dataset**

<table>
<thead>
<tr>
<th>Network</th>
<th>Nottingham Piano-midi</th>
<th>Muse</th>
<th>JSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recurrent</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Dense</td>
<td>512 512,512</td>
<td>512,512</td>
<td>256,256</td>
</tr>
<tr>
<td>Emission</td>
<td>1536</td>
<td>1536</td>
<td>1536</td>
</tr>
<tr>
<td>Transition</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
</tbody>
</table>

*Comma-separated values represent multiple hidden layers, e.g., for Piano-midi, the model has two layers in its Dense network

*All cases except JSB use 16 LSGM states; JSB uses 32 LSGM states